R. Sasse
Ch. Sprenger
SS 2019

Formal Methods for Information Security

Project Assignment

Due date: Wednesday, May 29, 2019, 12:00, via e-mail

In this project, you will develop two different key exchange protocols by a series of successive (informal) refinement steps. The first protocol is Password-Authenticated Connection Establishment (PACE), which is part of the protocol suite used for machine-readable travel documents (e.g., e-passports) all over the world. The second protocol is the Off-the-Record (OTR) protocol, which is used for secure messaging. Carefully read the instructions below before you begin.

Instructions

The protocol assignment has two parts. In each part, you will create several protocol models capturing different aspects of the PACE and OTR protocols. Solve both parts of the assignment given below, i.e., model the different parts of the PACE and OTR protocols. Your results will be Tamarin theory files and a descriptive report.

Here is a list of general instructions:

Teamwork The project should be completed by pairs of students working together. Please find a partner and let us know who you are working with by email to both lecturers. Do this by March 28 at the latest.

Theory files Ensure that the file name and the theory name do match. Store theory *X* in file *X.spthy*, i.e., that file starts with `theory X begin`. Check that each of your protocol specifications can be loaded without any error or warning. Annotate each lemma with a comment indicating whether it gets verified or not.

Executability Ensure that each model is executable. To this end, write an executability lemma, prove it, and check that it is the expected execution without adversary involvement. You may have to find a proof manually, or write a more precise lemma, to get the expected execution.

Tagging It may be helpful to add tagging information to your messages so that the participants can distinguish them more easily if you find that necessary. For example, the initiator could add the string 'initiator' to messages it sends.

Modeling guideline Write the protocol model such that messages are exchanged in the chronological order, i.e., alternate send and receive rules so that the receive following a send rule accepts exactly what was sent before (assuming no adversary-involvement). This makes it easier to convince yourselves you have a correct model.

Security properties For secrecy and (non-)injective agreement you **must** use the lemmas as defined in the lecture slides but may adjust them to separate agreement from the different viewpoints of initiator (respectively responder), given in the appendix for your convenience. Thus, you must annotate your protocol rules with the appropriate action labels yourself as described in the lecture.

Properties from all points of view Do ensure that you look at the security properties not just for the initiator or the responder, but for each of them.

Describing attacks We require that you describe attacks in your report as found by your handed-in Tamarin models. Make sure that your description covers all steps of the attack given, e.g., as Alice&Bob notation or message sequence chart. Do not describe the attack you want to find, when the tool finds a different one! To have the tool find the attack you want, you can modify (weaken) the lemma to exclude the unwanted attack. Should you find that your modified lemma is verified, you have weakened your lemma too much and must undo such changes. Alternatively, you can also add restrictions for similar effect (see exercise sessions for this, or ask). Ensure that when you go to the next model you start with the unmodified standard properties (particularly for 2.2 to 2.3).

No communication partner excluded Agents may talk to themselves. This is not something to generally remove. This is the basis of some possible reflection attacks.

Grading Each assignment (1.1-1.5 and 2.1-2.4) gives a maximum of 10 points (split between the theory and the description), and the overall quality (conciseness, consistency and clarity) of the report gives an additional maximum of 20 points, for a total of 110 points.

Getting help Do ask us questions if you are experiencing problems with the modeling or the Tamarin tool that you are unable to resolve yourself.

Deliverables

The deliverables consist of Tamarin theories and a *report* (in PDF format). The deliverables are specified in detail below. Email the deliverables as a zip or compressed tar archive by the due date to both instructors.

D1. Report The report should correspond to a “lab journal” recording all findings and containing the answers to the questions. The report should reference your protocol models in an un-ambiguous way, preferably according to the file names given in the questions. The report also **must** include your intended protocol in Alice&Bob specification for each question, i.e., one Alice&Bob specification for each theory.

D2. Theory files Hand in the security protocol theory (spthy) files of all relevant protocol models (i.e., Tamarin theories) for each assignment. Do not forget to ensure that all theories load without errors and warnings and to annotate each lemma with a comment indicating whether it gets verified or not.

D3. Proof files Hand in the proofs for all lemmas. You can obtain the proof files after having proved all lemmas by clicking the "Download" button in the upper-right corner of your interactive Tamarin session. Name the proof files *X-proof.spthy* for each theory *X*.

1. PACE protocol

The PACE protocol is part of a protocol suite that is used for machine-readable travel documents all over the world. These travel documents include electronic passports and identity cards. The PACE protocol is used to establish a secure channel between the terminal and the RFID chip on the passport (or ID card). The protocol assumes that they initially only share a low-entropy secret and uses a Diffie-Hellman key exchange to derive a strong session key. In practice, the low-entropy pre-shared secret may be either from a six-digit Card Access Number (CAN) or the Machine-Readable Zone (MRZ), which is a code imprinted on the document and optically scanned by the terminal.

In this part of the project, we develop the PACE protocol by a series of successive refinements. We start from a very simple challenge-response protocol for which we show an agreement property. With each refinement, we introduce additional features and possibly additional or stronger security properties. Our goal is to derive PACE and prove that it satisfies perfect forward secrecy for its session key as well as mutual agreement of the parties on the session key and other protocol elements.

Assignment 1.1. A simple challenge-response protocol

Our initial protocol is the following simple MAC-based challenge response protocol P1 between an initiator A and a responder B.

$$\begin{array}{ll} A \to B: & x \\ B \to A: & [x]_{\mathsf{k}(A,B)} \end{array}$$

where x is a nonce generated by A, $[M]_K$ denotes the MAC of message M with key K, and $\mathsf{k}(A,B)$ is a symmetric long-term key shared by A and B. Make sure that the key $\mathsf{k}(A,B)$ is uni-directional, i.e., $\mathsf{k}(A,B) \neq \mathsf{k}(B,A)$.

Formalize this protocol in Tamarin and prove that it satisfies the authentication property that A injectively agrees with B on the nonce x. Make sure that the intruder can also act as a regular protocol participant. This is theory *P1*.

Assignment 1.2. Mutual authentication

The first refinement, protocol P2, consists of a combination of two instances of P1, one initiated by A (generating and sending nonce x) and the other one initiated by B (generating and sending nonce y). These two instances are run in an interleaved manner such that the senders alternate and the resulting protocol has four messages. The property we would like to achieve is the agreement of each role with the other role on *both* nonces x and y.

(a) Model the protocol P2 and analyze its desired security property in Tamarin as theory *P2a*. Can you find an attack?

(b) If you have found an attack: describe the problem, propose a fix as theory *P2b*, and try to prove the mutual injective agreement property. Iterate if necessary until you succeed with a proof.

Assignment 1.3. Introducing a session key

In the second refinement, we introduce a session key and a corresponding secrecy property. Instead of using the long-term key $\mathsf{k}(A, B)$ for macing, we mac in both directions with the session key Kab, which is derived from the long-term key and the nonces x and y using a key derivation function kdf as follows:

$$Kab = \mathsf{kdf}(\mathsf{k}(A, B), x, y)$$

At the same time, instead of macing both nonces, we only mac the other role's nonce (i.e., A macs y and B macs x).

(a) Define the modified protocol in Tamarin as theory *P3a* and prove mutual agreement on x, y, and Kab as well as the secrecy of Kab. (You can declare the function kdf as a user-defined function in Tamarin.)

(b) Give an informal justification of why we can include each role's own nonce in the agreement despite that nonce not being maced. Compare with protocol P2.

Assignment 1.4. Replace the password by a nonce

In this refinement, we replace the (low-entropy) password $\mathsf{k}(A, B)$ in the session key Kab by a (high-entropy) nonce s generated by A, i.e.,

$$Kab = \mathsf{kdf}(s, x, y).$$

Moreover, we add the symmetric encryption of s with the hashed password $\mathsf{h}(\mathsf{k}(A, B))$ as a second (plaintext) component to the first message from A to B. We call this protocol P4.

Model the protocol P4 in Tamarin as theory *P4* and show that it satisfies the same properties as the protocol P3.

Assignment 1.5. Introducing Diffie-Hellman: The PACE protocol

In the final refinement, protocol P5, we derive the PACE protocol by replacing P4's nonces x and y by Diffie-Hellman half-keys g^x and g^y. A particularity of the PACE protocol is the choice of the generator g, which is defined by

$$g = \mathsf{map}(s, p).$$

Here, s is the nonce from P4 and p is a public domain parameter that, together with the mapping function map, ensures that g is a suitable group generator. We model map as a user-defined hash function (i.e., without equational properties). The parameter p is added as an additional plaintext component to the first message. Furthermore, the hashed Diffie-Hellman secret

$$Kab = \mathsf{h}(g^{xy}).$$

now replaces the previous session key.

(a) Transform your model of P4 into a model of the PACE protocol as described above and establish the same properties as for P4 (modulo the replacement of nonces by half-keys) as theory *P5ab*.

Hint: Use "let $g = \mathsf{map}(s, p)$" in each protocol rule instead of storing g in state facts.

(b) Strengthen the secrecy property of the session key Kab to perfect forward secrecy. Add this property to theory *P5ab*.

(c) Explain how the protocol relies on the secrecy of the base g for the exponentiation. Would it still work with a public base? Justify your answer.

(d) Remove any tags that you may have in your protocol so that the last two messages become unifiable. Find an attack on the secrecy or authentication property of the resulting protocol and fix it by ensuring that g^x and g^y differ.[1] This check can be realized in Tamarin using a restriction. Specify this as theory *P5d*.

[1] Having unifiable messages in a protocol tends to make a protocol prone to attacks, but this is what PACE does.

2. The Off-the-Record Messaging Protocol

This project aims to model the key exchange phase of the Off-the-Record Messaging (OTR) protocol. It is based on the paper [1]. The OTR protocol consists of two phases.

Phase 1 is an authenticated key-exchange protocol. At the end of phase 1, the communicating parties have a shared session key.

Phase 2 is the messaging protocol in which the parties continuously refresh the session key while exchanging their messages.

In the following we will consider the key-exchange part (phase 1) of the OTR protocol.

Assignment 2.1. Modeling the original OTR Key Exchange

Model the OTR Authenticated Key-Exchange shown in Section 2.1 of [1] as theory *OTR1* and informally justify its correctness. To justify correctness, explain how your model relates to the protocol description given in [1].

Assignment 2.2. Authentication Failure

Demonstrate the authentication failure described in Section 3.1 of [1] on the model you obtained in the first step. Do this by specifying a suitable authentication lemma and exhibiting an attack trace as theory *OTR2*. (Note: This lemma is allowed to be non-standard, “Describing attacks” in the instructions on page 2.)

Ensure (by manual inspection) that the attack trace you store is the one from the paper [1]!

Assignment 2.3. Improvement

Improve the protocol as suggested in Section 3.1 and analyze the standard authentication lemma (for both points of view) as theory *OTR3*.

Assignment 2.4. SIGMA

Improve the protocol further by modeling the “SIGMA” authenticated key exchange (AKE) protocol proposed in Section 4 of [1] as theory *OTR4*.

References

[1] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Secure Off-the-Record Messaging. In *WPES*, pages 81–89, 2005.

A. Lemmas

```
lemma secrecy:
  "All A x #i.
    Secret(A,x) @i ==>
    not (Ex #j. K(x)@j)
        | (Ex X #r. Reveal(X)@r & Honest(X) @i)"

lemma secrecy_PFS:
  "All A x #i.
    Secret(A,x) @i ==>
    not (Ex #j. K(x)@j)
        | (Ex X #r. Reveal(X)@r & Honest(X) @i & r < i)"

lemma noninjectiveagreementINITIATOR:
  "All a b t #i.
    Commit(a,b,<'I','R',t>) @i
    ==> (Ex #j. Running(b,a,<'I','R',t>) @j)
        | (Ex X #r. Reveal(X)@r & Honest(X) @i)"

lemma injectiveagreementINITIATOR:
  "All a b t #i.
    Commit(a,b,<'I','R',t>) @i
    ==> (Ex #j. Running(b,a,<'I','R',t>) @j
        & j < i
        & not (Ex a2 b2 #i2. Commit(a2,b2,<'I','R',t>) @i2
                           & not (#i2 = #i)))
              | (Ex X #r. Reveal(X)@r & Honest(X) @i)"


lemma noninjectiveagreementRESPONDER:
  "All a b t #i.
    Commit(a,b,<'R','I',t>) @i
    ==> (Ex #j. Running(b,a,<'R','I',t>) @j)
        | (Ex X #r. Reveal(X)@r & Honest(X) @i)"

lemma injectiveagreementRESPONDER:
  "All a b t #i.
    Commit(a,b,<'R','I',t>) @i
    ==> (Ex #j. Running(b,a,<'R','I',t>) @j
              & j < i
              & not (Ex a2 b2 #i2. Commit(a2,b2,<'R','I',t>) @i2
                                 & not (#i2 = #i)))
              | (Ex X #r. Reveal(X)@r & Honest(X) @i)"
```